

## SOLUTIONS TO HOMEWORK 2

### Section 2.3

18) For MatLab output, see the last few pages.

a) bisection = .447656

b) false position = .442066949

c) secant = -2989.9400

d) newton with  $p_0 = 0$  returns 13.6550122

newton with  $p_0 = .48$  returns .447431543

The actual value given in the book is .447431543. So Newton's method returns the exact nine digit approximation provided we start at the correct initial approximation. Starting with  $p_0 = 0$  has the same problem as the secant method. These diverge due to the wild behavior of this function in the interval in which we are seeking a root. There is a singularity at  $\frac{1}{2}$ . This singularity also causes problems for the method of false position. However, since the method of false position includes a bracketing step (keeping the root in between the next approximations) it is guaranteed to converge. In this case, it does not converge as quickly as the bisection section method (straight bracketing) because the bad behavior of the function causes the secant approximations in the method of false position to take small steps while the bisection method always the interval in half. The bisection method only considers the sign of the function value, not the function value itself. Therefore, the bisection method ignores the singularity.

28) a)  $C(t) = Ate^{-\frac{t}{3}}$  so  $C'(t) = Ae^{-\frac{t}{3}}(1 - \frac{t}{3})$ . Therefore,  $t = 3$  is the only time that a critical point occurs in this function. Therefore, we must find  $A$  such that  $C(3) = 1$ . We see that  $A = \frac{e}{3}$  and we have  $C(t) = \frac{e}{3}te^{-\frac{t}{3}} = \frac{t}{3}e^{1-\frac{t}{3}}$ .

b) We want to find  $t$  such that  $C(t) = \frac{1}{4}$ . Since we know that  $C(3) = 1$  we choose an initial approximation of 4. We perform Newton's method on the function  $f(t) = C(t) - .25$  with initial approximation  $t_0 = 4$ . My algorithm returns  $t^* = 11.0779$ . The first booster should be given at about 665 minutes (11 hours 5 minutes).

c) The initial dose was  $\frac{e}{3}$  so the first booster should be  $\frac{3}{4}\frac{e}{3} = \frac{e}{4}$ . Assuming an additive effect, and that the first booster was given at  $t = 11.0779$  hours, we write the function  $C_1(t)$  that defines the blood concentration of the drug at  $t$  hours after the initial dose. This function is only defined for  $t \geq 11.0779$ .

$$C_1(t) = C(t) + \frac{e}{4}(t - 11.0779)e^{-\frac{t-11.0779}{3}}.$$

For Newton's method, we need the derivative:

$$C'_1(t) = C'(t) + \frac{e}{4} \left[ \frac{14.0779 - t}{3} \right] e^{-\frac{t-11.0779}{3}}.$$

We wish to find when  $C_1(t) = \frac{1}{4}$  so we perform Newton's method on the function  $g(t) = C_1(t) - .25$ . Since  $g$  is decreasing on  $(14.0779, \infty)$  we can choose any appropriate initial approximation larger than say 15 and be sure we converge. Since the last decrease took 8 hours, let's use an initial approximation of  $t_0 = 19$ . From Newton's method, the next booster should be given at  $t^{**} = 21.233$  or at about 21 hours 14 minutes after the initial dose. That's 10 hours 9 minutes after the first booster.

Section 2.4

8) (a)  $p_n = 10^{-2^n}$  so  $(p_n)^2 = (10^{-2^n})^2 = 10^{-2^{n+1}} = p_{n+1}$ . We know  $p_n \rightarrow 0$  so  $p = 0$ . Therefore

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^2} = \lim_{n \rightarrow \infty} \frac{|p_{n+1}|}{|p_n|^2} = \lim_{n \rightarrow \infty} 1 = 1.$$

So by definition,  $p_n$  converges to 0 of order 2 with asymptotic error constant 1. Since the order is 2,  $p_n$  converges quadratically to 0.

(b)  $p_n = 10^{-n^k}$  so  $p_n^2 = 10^{-2n^k}$  and  $p_{n+1} = 10^{-(n+1)^k}$ . Again  $p = 0$ . Therefore

$$\frac{|p_{n+1} - p|}{|p_n - p|} = \frac{10^{-(n+1)^k}}{10^{-2n^k}} = 10^{2n^k - (n+1)^k}.$$

Now  $(n+1)^k = n^k + a_{k-1}n^{k-1} + \dots + 1 = n^k + q_{k-1}(n)$  where  $q_{k-1}(n)$  is a  $k-1$  degree polynomial in  $n$ . Therefore, regardless of what  $k \in \mathbb{Z}^+$  we choose,

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{|p_{n+1}|}{|p_n|} &= \lim_{n \rightarrow \infty} 10^{2n^k - (n+1)^k} \\ &= \lim_{n \rightarrow \infty} 10^{n^k - q_{k-1}(n)} = \infty. \end{aligned}$$

Hence,  $p_n$  can not converge to 0 quadratically for any choice of  $k$ .

G14) Since  $p_n \rightarrow p$  of order  $\alpha$ , then  $\lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda$  for some constant  $\lambda$ . For large  $n$  we have  $|p_{n+1} - p| \approx C |p_n - p| |p_{n-1} - p|$ . Also, since the sequence converges, for large  $n$  we know that  $|p_{n+1} - p| \approx |p_n - p|$ . This leads to the following:

$$\begin{aligned} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} &\approx \frac{C |p_n - p| |p_{n-1} - p|}{|p_n - p|^\alpha} \\ &= \frac{C |p_{n-1} - p|}{|p_n - p|^{\alpha-1}} \\ &\approx \frac{C |p_n - p|}{|p_{n+1} - p|^{\alpha-1}}. \end{aligned}$$

Using the properties of the limit, we now establish that  $p_n \rightarrow p$  of order  $\frac{1}{\alpha-1}$ :

$$\begin{aligned} \frac{1}{C^{\frac{1}{\alpha-1}}} \left( \lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^{\frac{1}{\alpha-1}}} \right) &= \left( \lim_{n \rightarrow \infty} \frac{1}{C} \frac{|p_{n+1} - p|^{\alpha-1}}{|p_n - p|} \right)^{\frac{1}{\alpha-1}} \\ &= \left( \lim_{n \rightarrow \infty} C \frac{|p_n - p|}{|p_{n+1} - p|^{\alpha-1}} \right)^{\frac{-1}{\alpha-1}} \\ &\approx \left( \lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} \right)^{\frac{-1}{\alpha-1}} \\ &= \lambda^{\frac{-1}{\alpha-1}}. \end{aligned}$$

Therefore, since  $p_n \rightarrow p$  of order  $\frac{1}{\alpha-1}$  and of order  $\alpha$ , it must be that  $\frac{1}{\alpha-1} = \alpha$  or that  $\alpha^2 - \alpha - 1 = 0$ . Since we know  $\alpha > 0$ , we take the positive root and have verified that  $\alpha = \frac{1+\sqrt{5}}{2}$ .

Section 2.5

8) Use Steffenson's method (ALG 2.6) to find the root of  $f(x) = x - 2^{-x}$  with a tolerance of  $10^{-4}$  and compare this to the fixed point iteration result from the previous homework.

My implementation of Steffenson's method with an initial approximation of .5 provides the approximation .6411857 in 3 iterations at the tolerance  $10^{-4}$ . On the previous homework, with the same tolerance, I obtain the approximation .6412053 in 11 iterations of the fixed point method. That's a massive speedup with approximately a 72% reduction in iterations. (Using the tic-toc feature of Matlab, I appear to get better than a 25% reduction in time.)

14) a) Suppose  $p_n \rightarrow p$  of order  $\alpha > 1$  with asymptotic error constant  $\lambda$ . Then, since  $\alpha - 1 > 0$  and  $|p_n - p| \rightarrow 0$ , we have

$$\begin{aligned} \lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|} &= \lim_{n \rightarrow \infty} \left( \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} \frac{1}{|p_n - p|^{1-\alpha}} \right) \\ &= \lambda \lim_{n \rightarrow \infty} \left( \frac{1}{|p_n - p|^{1-\alpha}} \right) \\ &= \lambda \lim_{n \rightarrow \infty} |p_n - p|^{\alpha-1} \\ &= 0 \end{aligned}$$

Thus, if  $p_n \rightarrow p$  of order  $\alpha > 1$ , then  $p_n$  converges to  $p$  superlinearly.

b) Let  $t_n$  simply denote the appropriate degree  $n$  polynomial in  $n$  defined by  $t_n = (n+1)^{n+1} - n^{n+1}$ .  $p_n = n^{-n}$  clearly converges to 0. Suppose  $\alpha \geq 1$ , then

$$\frac{|p_{n+1}|}{|p_n|^\alpha} = \frac{(n+1)^{-(n+1)}}{n^{-\alpha n}} = \frac{n^{\alpha n}}{(n+1)^{(n+1)}} = \frac{\frac{n^{\alpha n}}{n^{(n+1)}}}{1 + \frac{t_n}{n^{n+1}}}.$$

Therefore

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1}|}{|p_n|^\alpha} = \lim_{n \rightarrow \infty} \frac{n^{\alpha n}}{n^{(n+1)}} = \lim_{n \rightarrow \infty} n^{(\alpha-1)n-1}.$$

Suppose  $\alpha > 1$ . Then for all  $n > \frac{1}{\alpha-1}$ , the exponent is positive and the sequence grows. Therefore for all  $\alpha > 1$ ,  $\lim_{n \rightarrow \infty} \frac{|p_{n+1}|}{|p_n|^\alpha} = \infty$  and so  $p_n$  can not converge to 0 of order  $\alpha$ .

Now suppose  $\alpha = 1$ . Then from the last equation we have

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1}|}{|p_n|^\alpha} = \lim_{n \rightarrow \infty} n^{(\alpha-1)n-1} = \lim_{n \rightarrow \infty} n^{-1} = 0.$$

Therefore,  $p_n$  is superlinearly convergent to 0.

Section 3.1

6) c) We want to interpolate at the points  $(.1, -.29004986)$ ,  $(.2, -.56079734)$ ,  $(.3, -.81401972)$ , and  $(.4, -1.0526302)$ . Following the definition for a Lagrange polynomial, we create degree 1,2,3 interpolating polynomials and approximate  $f(.18)$ ,

$$P_1(x) = -.29004986 \frac{(x - .2)}{(-.1)} - .56079734 \frac{(x - .1)}{(.1)}$$

$$P_2(x) = -.29004986 \frac{(x - .2)(x - .3)}{(-.1)(-.2)} - .56079734 \frac{(x - .1)(x - .3)}{(.1)(-.1)} - .81401972 \frac{(x - .1)(x - .2)}{(.2)(.1)}$$

$$P_3(x) = -.29004986 \frac{(x - .2)(x - .3)(x - .4)}{(-.1)(-.2)(-.3)} - .56079734 \frac{(x - .1)(x - .3)(x - .4)}{(.1)(-.1)(-.2)} - .81401972 \frac{(x - .1)(x - .2)(x - .4)}{(.2)(.1)(-.1)} - 1.0526302 \frac{(x - .1)(x - .2)(x - .3)}{(.3)(.2)(.1)}$$

So our approximations are

$$P_1(.18) = -.29004986 \frac{(-.02)}{(-.1)} - .56079734 \frac{(.08)}{(.1)} = -.50664784$$

$$P_2(.18) = -.29004986 \frac{(-.02)(-.12)}{(-.1)(-.2)} - .56079734 \frac{(.08)(-.12)}{(.1)(-.1)} - .81401972 \frac{(.08)(-.02)}{(.2)(.1)} = -.50804984$$

$$P_3(.18) = -.29004986 \frac{(-.02)(-.12)(-.22)}{(-.1)(-.2)(-.3)} - .56079734 \frac{(.08)(-.12)(-.22)}{(.1)(-.1)(-.2)} - .81401972 \frac{(.08)(-.02)(-.22)}{(.2)(.1)(-.1)} - 1.0526302 \frac{(.08)(-.02)(-.12)}{(.3)(.2)(.1)} = -.50814310$$

8) c) Using my implementation of Neville's method to generate the polynomials, I get similar approximations (NP stands for Neville's method polynomial):

$$NP_1(.18) = -.50664784$$

$$NP_2(.18) = -.50804985$$

$$NP_3(.18) = -.50814307$$

10) c) Since  $f(x) = x^2 \cos(x) - 3x$  we can compute that  $f''(x) = (2 - x^2) \cos x - 4x \sin x$  and  $f'''(x) = (x^2 - 6) \sin x - 6x \cos x$ . We use the bounds from the Taylor series  $|\sin x| \leq |x|$  and the usual  $|\cos x| \leq 1$  so that  $|f''(\xi)| \leq |2 - \xi^2| + |4\xi^2|$ . Then

$$|P_1(.18) - f(.18)| = \frac{|f''(\xi(x))|}{2} |.08| |.02| \leq \frac{|2 - .1^2| + |4(.2)^2|}{2} |.08| |.02| = .00172.$$

The actual error is  $|- .50664784 + .50812346| = .00148$  so we are exceeding the theoretical bound.

For the degree two polynomial, we perform precisely the same analysis, although now we need  $f'''(x) = (x^2 - 6) \sin x - 6x \cos x$  so that  $|f'''(\xi)| \leq |\xi^2 - 6| |\xi| + |6\xi|$ . Therefore

$$|P_2(.18) - f(.18)| \leq \frac{(.08)(.02)(.12)}{6} (|.1^2 - 6| .3 + 6|.3|) \approx .0001151.$$

The actual absolute error was  $|-.50804984 + .50812346| = .00007362$ .

### Section 3.2

8) Using my implementation of Newton's Divided Difference Algorithm (ALG3.2) (see MatLab workspace for output)

a) Using five points, we get a fourth degree polynomial. We'll call this  $Q(x)$ :

$$Q(x) = -6 + 1.0517x + .5725x(x - .1) + .2150x(x - .1)(x - .3) + .0630x(x - .1)(x - .3)(x - .6).$$

b) Adding another point, adds another row. So the forward difference method simply adds another term. This fifth degree polynomial is:

$$T(x) = Q(x) + .0142x(x - .1)(x - .3)(x - .6)(x - 1).$$

### Section 3.3

6)  $f(x) = 3xe^x - e^{2x}$  so that  $f^{(k)}(x) = 3(x+k)e^x - 2^k e^{2x}$ . The MatLab output on the following pages provide our approximations. Using the derivatives and the known intervals we get a bound. For the first approximation, we have to bound the fourth derivative on  $[1, 1.05]$ , so  $|f^{(4)}(\xi)| \leq |f^{(4)}(1.05)| \approx 87.3653$ . Therefore

$$|H_3(x) - f(x)| \leq 87.3653 \frac{|x-1|^2 |x-1.05|^2}{4!}$$

$$|H_3(1.03) - f(1.03)| \leq 87.3653 \frac{(.03)^2 (.02)^2}{24} = 1.3105 \times 10^{-6}.$$

On the interval  $[1, 1.07]$  we have a bound on  $f^{(6)}$  with  $|f^{(6)}(\xi)| \leq |f^{(6)}(1.07)| \approx 482.1288$ . So we have

$$|H_5(x) - f(x)| \leq 482.1288 \frac{|x-1|^2 |x-1.05|^2 |x-1.07|^2}{6!}$$

$$|H_5(1.03) - f(1.03)| \leq 482.1288 \frac{(.03)^2 (.02)^2 (.04)^2}{720} = 3.8570 \times 10^{-10}.$$

The actual value is  $f(1.03) = .80932362$  and our implementation of the Hermite polynomials returned the approximations  $H_3(1.03) = .80932486$  and  $H_5(1.03) = .80932362$ . Therefore (using variable precision arithmetic) the actual error for  $H_3$  is  $1.2373 \times 10^{-6}$  and for  $H_5$  it is  $3.6101 \times 10^{-10}$ . The error of the actual approximations are very close to the theoretical approximation of the error. Since we are dealing with a continuously differentiable function on a very small interval, it is reasonable that our Hermite interpolation is returning an error that is essentially equal to the theoretical upper bound for the error..

Matlab output for 2.3: 18.

```
>> vpa(BLAfalseposition(0,.48))
```

```
p =
```

```
0.1812  
0.2862  
0.3490  
0.3871  
0.4103  
0.4246  
0.4333  
0.4387  
0.4421
```

```
ans =
```

```
.442066949
```

```
>> vpa(BLABisect(0,.48))
```

```
p =
```

```
0.2400  
0.3600  
0.4200  
0.4500  
0.4350  
0.4425  
0.4462  
0.4481  
0.4472  
0.4477
```

```
y =
```

```
Final approximation is 0.447656.
```

```
>> vpa(BLAsecant(0,.48))
```

```
p =
```

```
0.1812  
0.2862  
1.0920  
-3.6923  
-22.6006  
-57.2228  
3.5388  
-113.9444  
-195.8950  
-2.9899e+003
```

```
ans =
```

```
-2989.94003753144534
```

```
>> vpa(BLAnewton(0))
```

```
p =  
1.9099  
3.7560  
4.9060  
7.8135  
9.2860  
9.8714  
11.6014  
11.8835  
13.6550  
ans =
```

```
13.655012218324751316345100349281
```

```
>> vpa(BLAnewton(0.48))
```

```
p =  
0.4676  
0.4551  
0.4486  
0.4475  
0.4474  
0.4474  
0.4474  
0.4474  
0.4474  
0.4474  
ans =
```

```
.44743154328874656933123787894147
```

MATLAB workspace for 3.2: 8.

```
>> xx=[0 .1 .3 .6 1];  
>> yy=[-6 -5.89483 -5.65014 -5.17788 -4.28172];  
>> DiffCoeff(xx,yy)
```

-4 sample points in a vector  
-function values  
-my code takes two vectors

ans =

-6.0000	0	0	0	0
-5.8948	1.0517	0	0	0
-5.6501	1.2234	0.5725	0	0
-5.1779	1.5742	0.7015	0.2150	0
-4.2817	2.2404	0.9517	0.2780	0.0630

```
>> xxx=[0 .1 .3 .6 1 1.1];  
>> yyy=[-6 -5.89483 -5.65014 -5.17788 -4.28172 -3.99583];  
>> DiffCoeff(xxx,yyy)
```

-add the new values

ans =

-6.0000	0	0	0	0	0
-5.8948	1.0517	0	0	0	0
-5.6501	1.2234	0.5725	0	0	0
-5.1779	1.5742	0.7015	0.2150	0	0
-4.2817	2.2404	0.9517	0.2780	0.0630	0
-3.9958	2.8589	1.2370	0.3566	0.0786	0.0142

Look, adding a new point just adds a new row to the matrix. You can just use the last polynomial plus this new coefficient times the product of  $(x-x_i)$  for all the previous points.

MATLAB workspace for 3.3: 6.

```
>> aa=[1 1.05];
>> ff=@(x) 3.*x.*exp(x)-exp(2.*x);
>> df=@(x) 3.*(x+1).*exp(x)-2.*exp(2.*x);
>> HermiteCoeff(aa, ff(aa), df(aa))
ans =
```

-set the sample points in a vector  
-write the function  
-the derivative of the function  
-my code for computing the coeff.

```
0.7658    0    0    0
0.7658  1.5316    0    0
0.8354  1.3928 -2.7749    0
0.8354  1.2422 -3.0124 -4.7502
```

```
>> Hermite(1.03,aa,ff(aa),df(aa))
ans =
```

-I wrote one to return the appx

```
0.8093
```

```
>> aaa=[1 1.05 1.07];
>> HermiteCoeff(aaa, ff(aaa), df(aaa))
ans =
```

-add the new point  
-rerun

```
0.7658    0    0    0    0    0
0.7658  1.5316    0    0    0    0
0.8354  1.3928 -2.7749    0    0    0
0.8354  1.2422 -3.0124 -4.7502    0    0
0.8589  1.1750 -3.3621 -4.9955 -3.5043    0
0.8589  1.1056 -3.4671 -5.2492 -3.6238 -1.7078
```

```
>> Hermite(1.03,aaa,ff(aaa),df(aaa))
ans =
```

-the appx is the same...really?

```
0.8093
```

```
>> vpa(Hermite(1.03,aa,ff(aa),df(aa)))
ans =
```

-if I use 8 digit rounding I get  
different answers

```
.80932486
```

```
>> vpa(Hermite(1.03,aaa,ff(aaa),df(aaa)))
ans =
```

```
.80932362
```